EXASCALE COMPUTING PROJECT
Preparing PETSc/TAO for Exascale

libEnsemble: A library for the concurrent evaluation of ensembles of computations

David Bindel[1,2] Stephen Hudson[1] Jeffrey Larson[1] Stefan M. Wild[1]
[1]Argonne National Laboratory [2]Cornell University

Argonne NATIONAL LABORATORY

Overview

libEnsemble is a Python library to coordinate the concurrent evaluation of ensembles of computations. Designed with flexibility in mind, libEnsemble can utilize massively parallel resources to accelerate the solution of design, decision, and inference problems.

libEnsemble aims for:

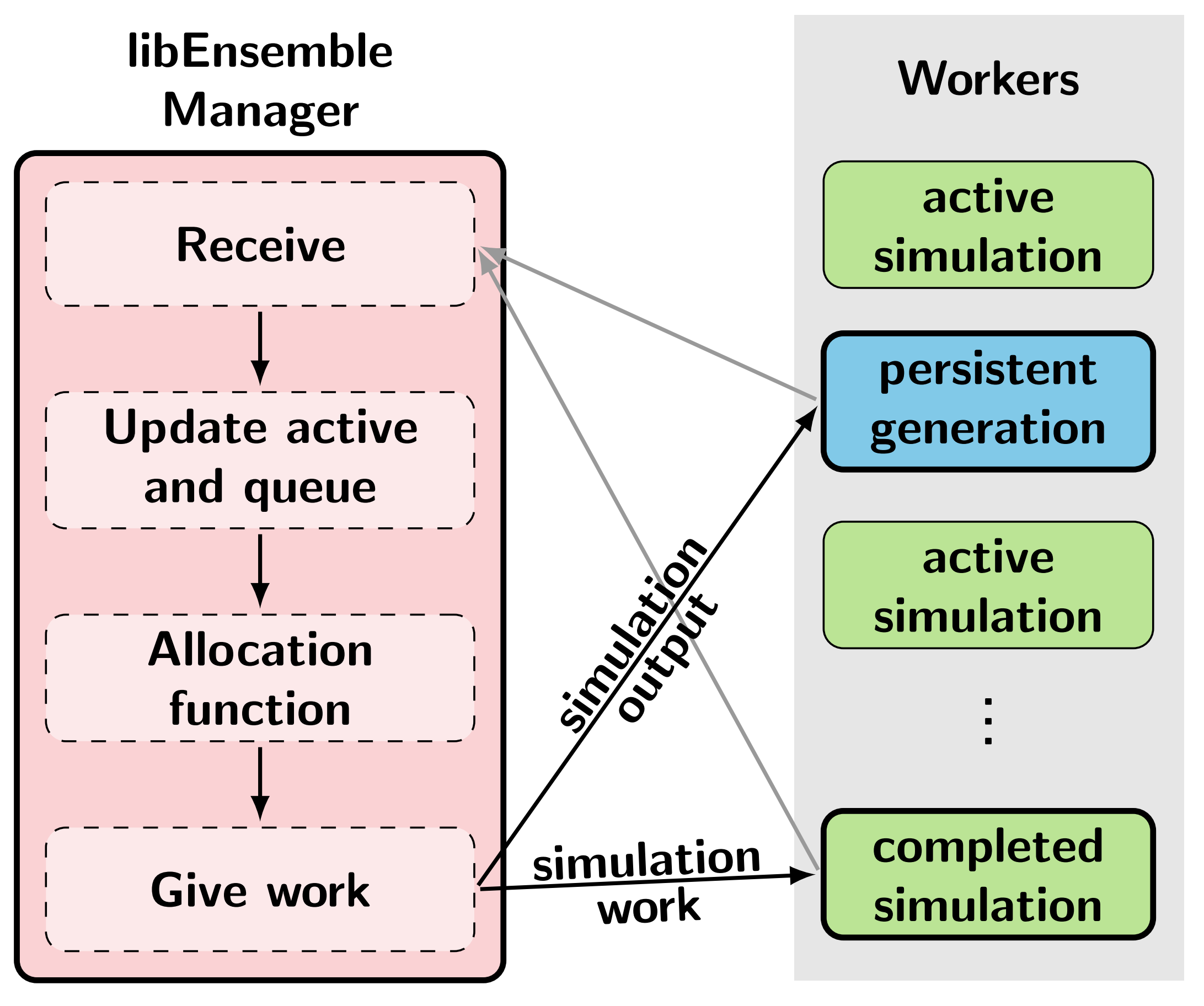
- Extreme scaling
- Fault tolerance
- Monitoring/killing jobs (recovers resources)
- Portability and flexibility
- Exploitation of persistent data/control flow

Manager and Workers

libEnsemble employs a manager-worker scheme that can run on various communication mediums (including MPI, multiprocessing, and TCP).

Workers can run simulation functions or generator functions (which create new parameters/inputs for simulations).

- Ex.- Random Sample $\rightarrow$ Simulations $\rightarrow$ Optimization $\rightarrow$ Simulations

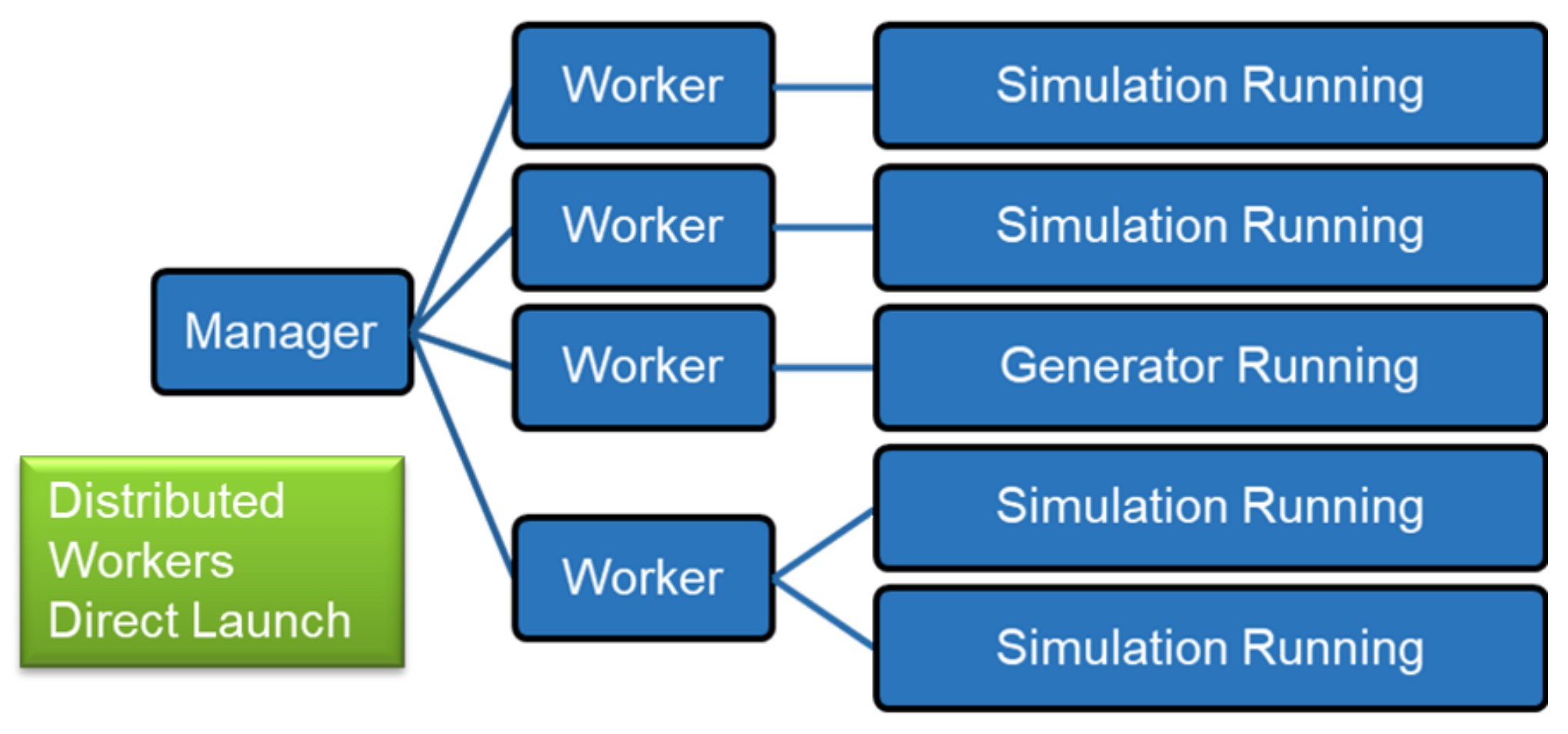


libEnsemble encapsulates the job control and worker communication layers making it highly adaptable to future systems and easy to experiment.

Flexible Run Mechanisms

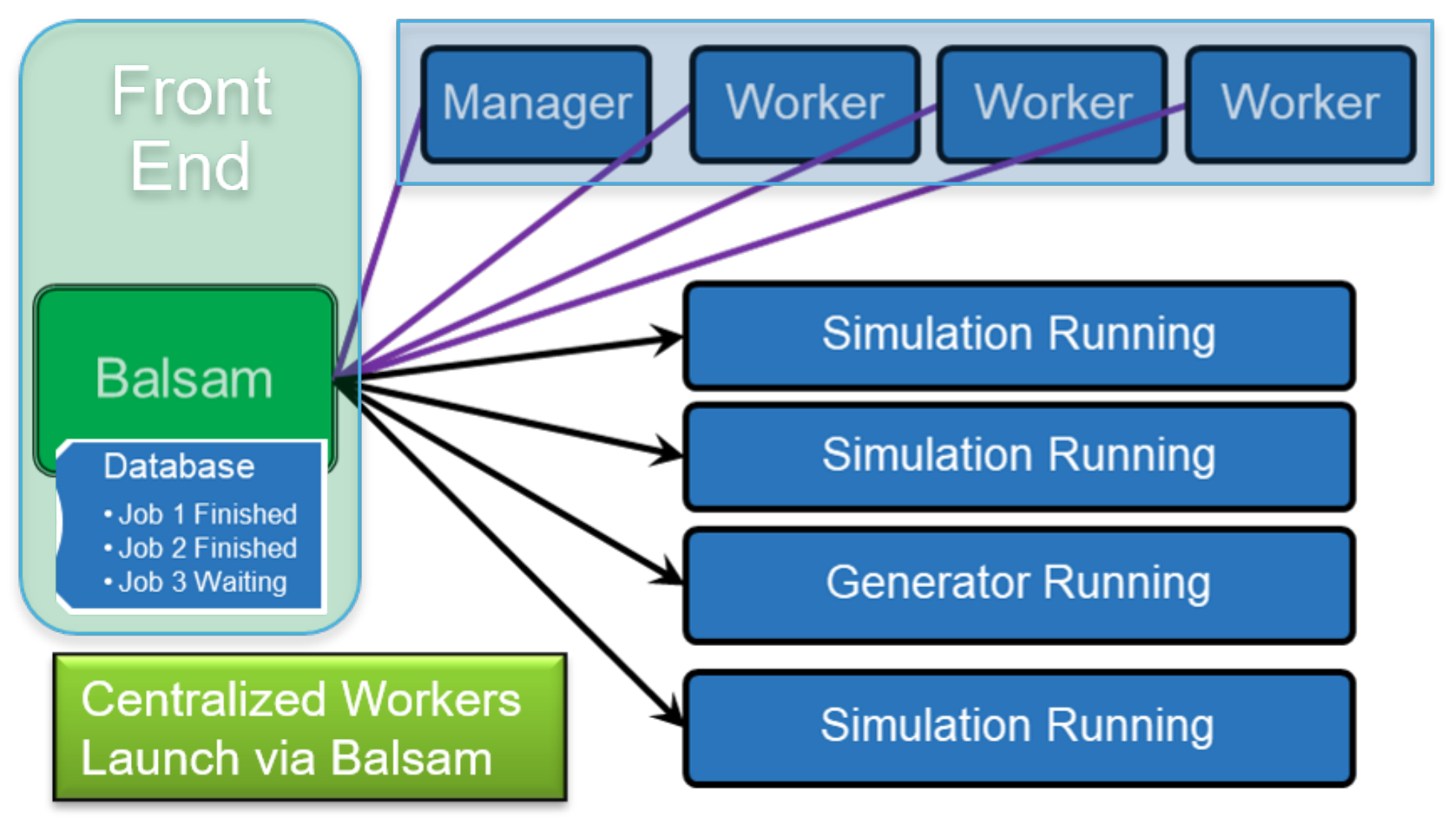
Workers can run in various configurations.

Distributed: Workers can run on compute nodes and launch jobs directly in-place.



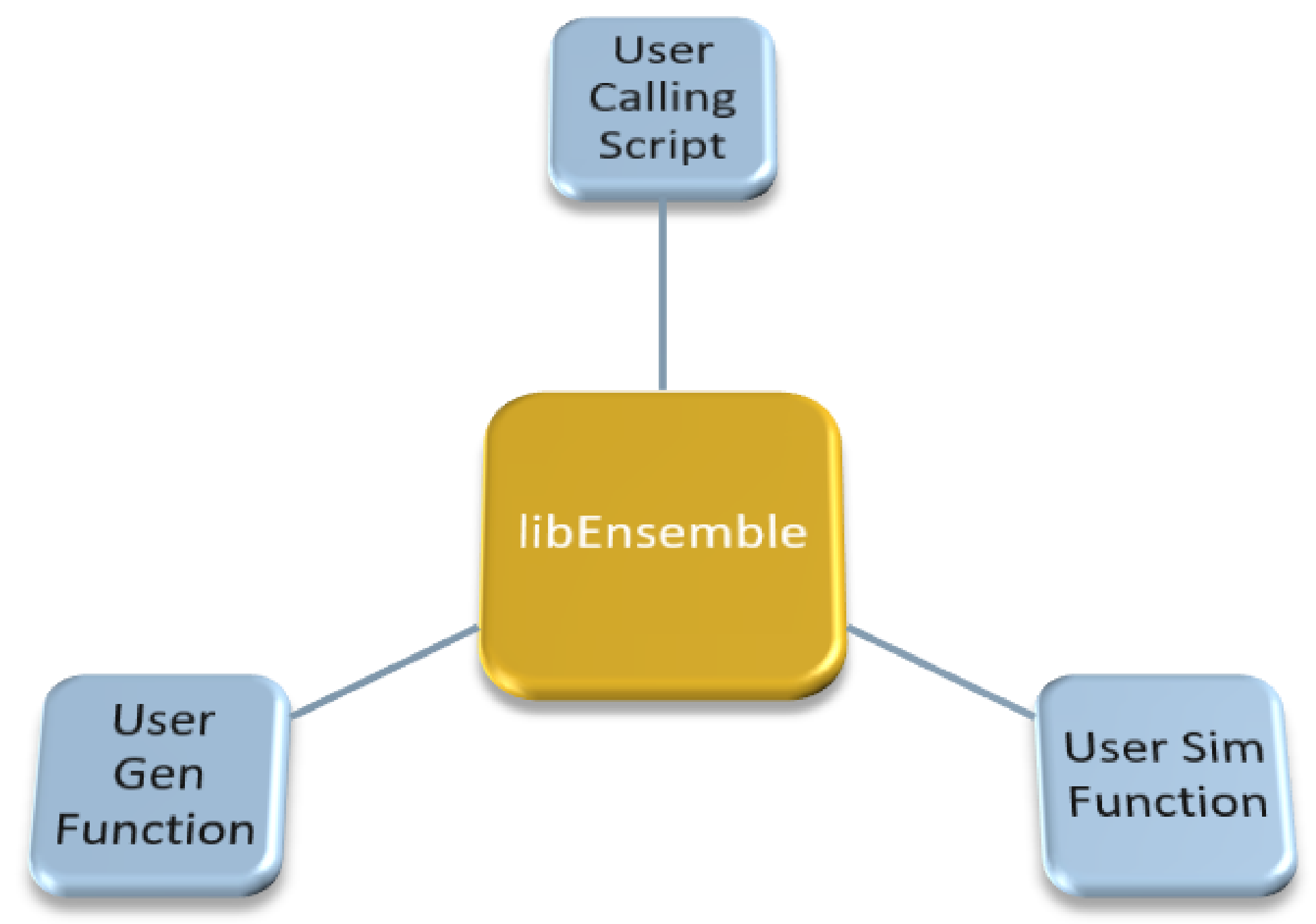
Centralized: Workers run on dedicated nodes and launch jobs to another set of nodes.

Balsam: Used as a proxy job launcher; Balsam runs on front end and maintains a database of jobs.



Using libEnsemble

The user selects or supplies a generation function that produces simulation inputs and a simulation function that performs and monitors the simulations. Several examples/templates of these functions are packaged with the library.

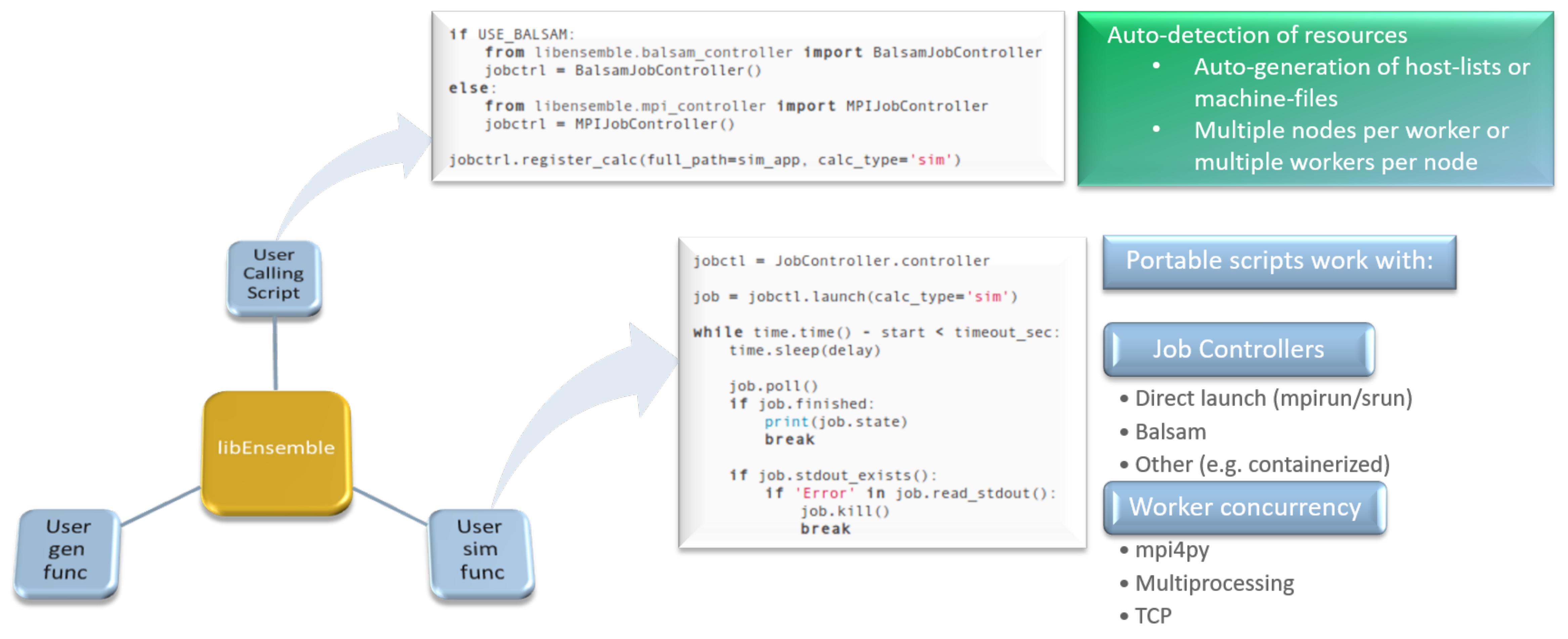


There are many potential use cases including:

Example gen. funcs	Example sim. funcs
• Bayesian parameter estimation	• Particle accelerator simulation
• Surrogate models	• Subsurface flow
• Sensitivity analysis	• PETSc simulations
• Design optimization	• DFT calculations
• Supervised learning	• Quantum chemistry

Job Controller

A job controller interface is provided and allows users to write portable simulation/generator functions in Python. These are agnostic of both the job launch/management system and worker concurrency. The main job controller functions are *launch*, *poll*, and *kill*.



```
if USE_BALSAM:
    from libensemble.balsam_controller import BalsamJobController
    jobctrl = BalsamJobController()
else:
    from libensemble.mpi_controller import MPIJobController
    jobctrl = MPIJobController()

jobctrl.register_calc(full_path=sim_app, calc_type='sim')
```

```
jobctl = JobController.controller

job = jobctl.launch(calc_type='sim')

while time.time() - start < timeout_sec:
    time.sleep(delay)

    job.poll()
    if job.finished:
        print(job.state)
        break

    if job.stdout_exists():
        if 'Error' in job.read_stdout():
            job.kill()
            break
```

Auto-detection of resources
- Auto-generation of host-lists or machine-files
- Multiple nodes per worker or multiple workers per node

Portable scripts work with:

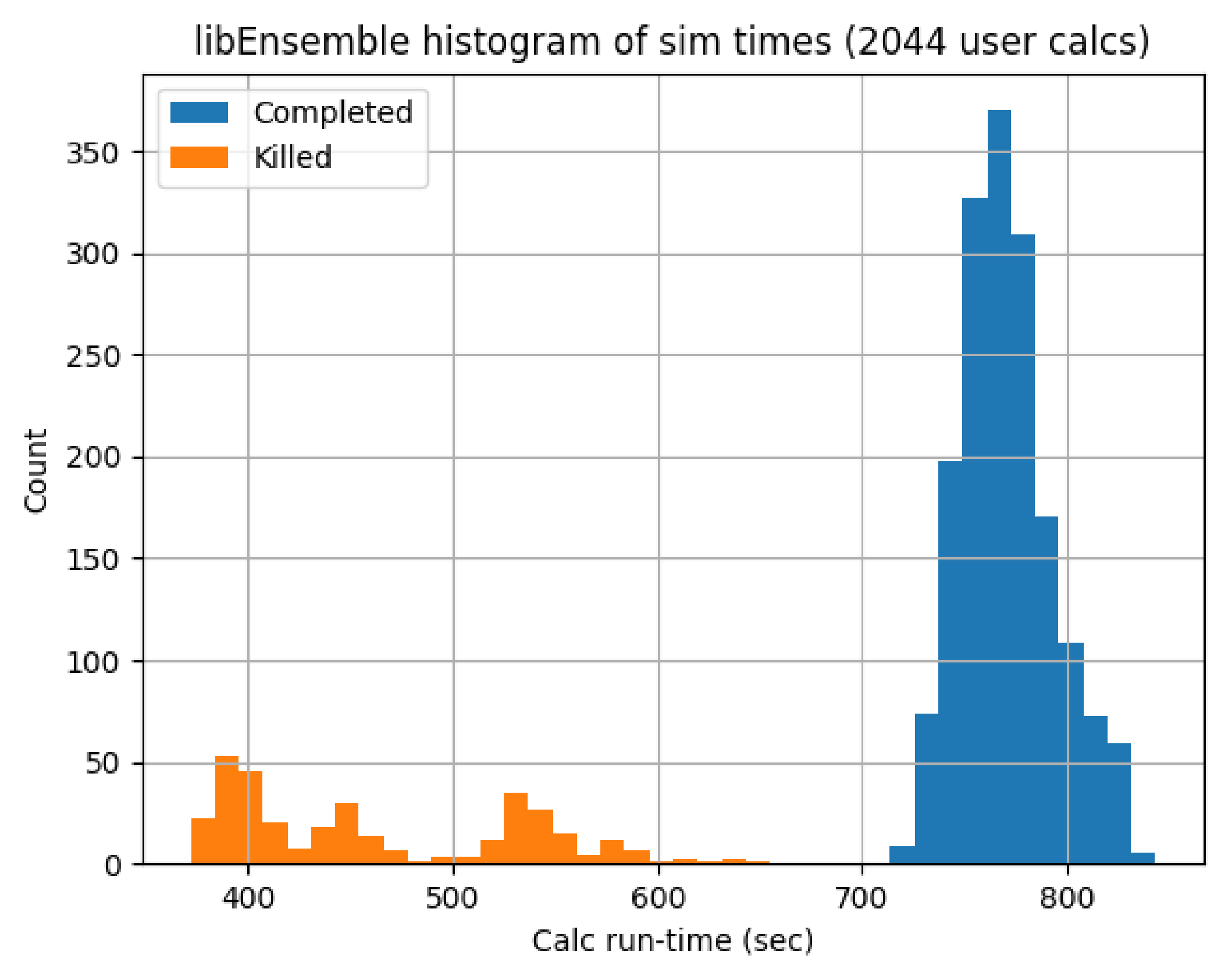
Job Controllers
- Direct launch (mpirun/srun)
- Balsam
- Other (e.g. containerized)

Worker concurrency
- mpi4py
- Multiprocessing
- TCP

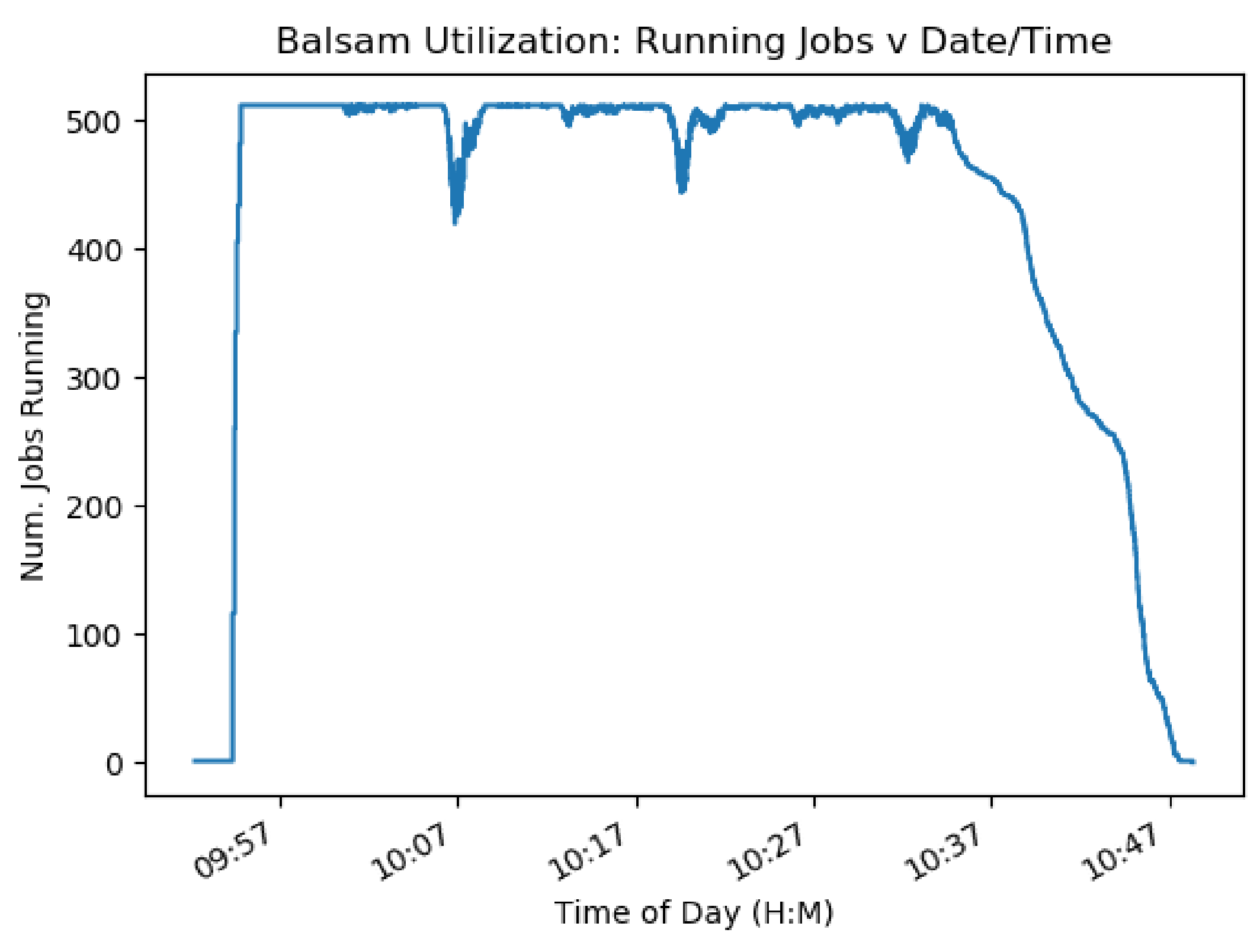
Running at Scale

Ex.- libEnsemble scaling: 1,030 node ensemble on ALCF/Theta (Cray XC40) with Balsam

- 511 workers (2,044 2-node simulations) using MPI
- OPAL (Object Oriented Parallel Accelerator Library) simulation functions



Histogram of completed and killed simulations (binned by run time). Killing jobs once they are identified as redundant improves efficiency of ensembles.



Total number of Balsam-launched applications running over time. The startup delay is due to parallel imports of Python libraries.

Future

Experimental/aspirational features:

- TCP communicator could allow workers to be dynamically added from cloud resources
- Job controllers can capitalize on novel features such as containerized launches to optimize resource partitioning and startup costs
- Persistent gen./sim. functions and distributed workers can make use of on-node resources (e.g., SSDs, GPUs)

Applications & Users Wanted!

libEnsemble is an open-source PETSc project:

`https://github.com/libensemble/libensemble`
`https://libensemble.readthedocs.io`

libEnsemble